Dart – Loop Control Statements (Break and Continue)

Dart supports two types of loop control statements:

1. Break Statement
2. Continue Statement

Break Statement:

This statement is used to break the flow of control of the loop i.e if it is used within a loop then it will terminate the loop whenever encountered. It will bring the flow of control out of the nearest loop.

Syntax:

```
break;
```

Example 1: Using break inside while loop

```
void main()
{
    int count = 1;

    while (count <= 10) {
        print(count);
        count++;

        if (count == 4) {
            break;
        }
    }
    print("Samir, you are out of while loop");
}
```

Output:

```
1
2
3
Samir, you are out of while loop
```

Explanation:

Initially count value is 1, as it goes inside loop the condition is checked, 1 <= 10 and as it is **true** the statement is printed variable is increased and then condition is checked, 2 == 4, which is **false**. Then the loop is followed again till the condition 4 == 4 is encountered and the flow comes out of the loop and then last print statement is executed.

Example 2: Using break inside do..while loop

```
void main()
{
    int count = 1;

    do {
        print(count);
        count++;

        if (count == 5) {
            break;
        }
    } while (count <= 10);
    print("Samir, you are out of do..while loop");
}
```

Output:

```
1
2
3
```

```
4
Samir, you are out of do..while loop
```

Example 3: Using break inside for loop

```
void main()
{
    for (int i = 1; i <= 10; ++i) {
        if (i == 2)
            break;

        print(i);
    }

    print("Samir, you are out of loop");
}
```

Output:

```
1
Samir, you are out of loop
```

Continue Statement:

While the **break** is used to end the flow of control, **continue** on the other hand is used to continue the flow of control. When a continue statement is encountered in a loop it doesn't terminate the loop but rather jump the flow to next iteration.

Syntax:

```
continue;
```

Example 1: Using continue inside while loop

```
void main()
{
    int count = 0;

    while (count <= 10) {
        count++;
        if (count == 4) {
            print("Number 4 is skipped");
            continue;
        }

        print(count);
    }

    print("Samir, you are out of while loop");
}
```

Output:

```
1
2
3
Number 4 is skipped
5
6
7
8
9
10
11
Samir, you are out of while loop
```

Explanation:

Here control flow of the loop will go smooth but when count value becomes 4 the if condition becomes true and the below statement is skipped because of continue and next iteration skipping number 4.

Example 2: Using continue inside do..while loop

```
void main()
{
    int count = 0;

    do {
        count++;

        if (count == 4) {
            print("Number 4 is skipped");
            continue;
        }

        print(count);
    } while (count <= 10);
    print("Samir, you are out of while loop");
}
```

Output:

```
1
2
3
Number 4 is skipped
5
6
7
8
9
10
11
Samir, you are out of while loop
```

Example 3: Using continue inside for loop

```
void main()
{
    for (int i = 1; i <= 10; ++i) {

        if (i == 2) {
            print(i);
            continue;
        }
    }

    print("Samir, you are out of loop");
}
```

Output:

```
2
Samir, you are inside loop
```

Labels in Dart

Most of the people, who have programmed in C programming language, are aware of **goto** and **label** statements which are used to jump from one point to other but unlike Java, Dart also doesn't have any *goto statements* but indeed it has **labels** which can be used with *continue* and *break* statements and help them to take a bigger leap in the code.

It must be noted that line-breaks are not allowed between **'label-name'** and loop control statements.

Example #1: Using label with the break statement

Dart

```
void main() {

  // Defining the label
  S1:for(int i=0; i<3; i++)
  {
    if(i < 2)
    {
      print("Samir, You are inside the loop");

      // breaking with label
      break S1;
    }
    print("You are still inside the loop");
  }
}
```

Output:

```
Samir, You are inside the loop
```

The above code results into only one-time printing of statement because once the loop is broken it doesn't go back into it.

Example #2: Using label with the continue statement

Dart

```
void main() {

  // Defining the label
  S1:for(int i=0; i<3; i++)
  {
    if(i < 2)
    {
      print("Samir, You are inside the loop");

      // Continue with label
      continue S1;
    }
    print("You are still inside the loop");
  }
}
```

Output:

```
Samir,  You  are  inside  the  loop
Samir,  You  are  inside  the  loop
You are still inside the loop
```

The above code results in printing of the statement twice because of the condition it didn't break out of the loop and thus printing it twice.